Available online at www.sciencedirect.com

ScienceDirect

Procedia Earth and Planetary Science 1 (2009) 1205–1211

Procedia Earth and Planetary Science

www.elsevier.com/locate/procedia

The 6th International Conference on Mining Science & Technology

Security management and deployment of distributed digital mine system

Zhang Ai-juan, Ji Cheng *

School of Computer Science & Technology ,China University of Mining & Technology, Xuzhou 221116, China

Abstract

Distributed system has been used in Digital Mine System. Because of the complexity inherent in distributed systems, security has become a crucial aspect in it. This paper proposed an access control mechanism for distributed system. By enforcing dynamic authorization as well as fine-grained organization of resource object, the access control policy provides the required flexibility of security management and deployment for Digital Mine System

Keywords: distributed system; authorization; fine-grained access control

1. Introduction

Distributed systems relying on middleware are used in Digital Mine System including smart cards, PDAs, embedded systems, standard PCs and workstations, and high–performance application servers. One of the prime reasons for employing distributed systems rather than centralized systems is the potential for combining resources to build systems that are more powerful than any single centralized system[1]. Another reason why distributed systems are used is the increased availability of resources that may be achieved if single point of failure is avoided and redundancy is provided in the design of a system.

An immediate consequence of all these uses of distributed systems is that multiple different nodes and resources have to be installed and managed. The complexity inherent in distributed systems constitutes a major problem for overall security.

2. Security problems and solutions

* Corresponding author. Tel.: +86-13852032203.
E-mail address: zaj@cumt.edu.cn.

1878-5220
doi:10.1016/j.proeps.2009.09.185

2.1. Security problems

The main question in distributed systems is how the specification, deployment and management of application–oriented access control policies in distributed object systems can be supported in a way that increases the overall security.

2.2. Ways to solve the problems

It is concluded that an integrated approach to secure software development and management is required and that it can best be supported by the definition of a declarative policy language. The technical feasibility of the access control is shown through an implementation of the required security infrastructure, which includes an interceptor–based access control mechanism, a language compiler, objects and role repositories.

Allocating security functions at the application level and not at the level of individual operating systems is first an important step to reduce the complexity incurred by heterogeneity. The primary focuses of Security Service are thus security policies for application objects. The central concepts of access control policy are views as a first–class concept for the type–safe aggregation of access rights, roles as a task–oriented abstraction of callers, and schemas as a means of specifying triggered dynamic changes in the protection state.

3. Access control mechanism

Any mechanism for controlling access to resource objects must be able to intercept and check all possible accesses, i.e., the mechanism must be interposed between the object and its callers and not be bypassed.

Interceptors are a convenient way to implement this mechanism and the security service specifies an access control interceptor for this purpose. The implementation presented here also uses an interceptor. Figure 1 illustrates how access requests are intercepted and then checked by an Access Decision object. If the access is allowed, it is passed on to the target object; otherwise a denial reply message is returned to the caller.

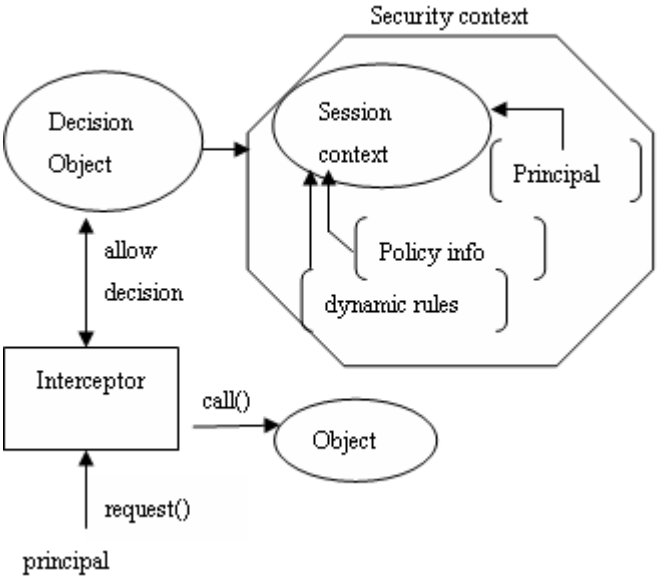

Fig. 1. Access control with interceptor

There is a session context for each principal-target pair. This session context contains the access policy information that applies to the target object, the principal information about the caller and a cache of previously made access decisions. The interceptor establishes this context upon the first request to a given target and the Access Decision object retrieves and updates it, if necessary.

Access control is performed when processes try to access protected objects at runtime. It is shown, however, that policy issues must be addressed in earlier stages in the application life cycle to enable appropriate runtime management. The following context describes the management tasks related to access policies at runtime and the deployment of earlier stages.

The three main tasks that can be identified correspond directly to what can be called the three basic dimensions of access control: principal management, object management and policy management, they and their relationship to access policy are examined as following.

3.1. Policy management

- Policy representation

The improved access control policy includes several concepts: subject(principal), role, view, object. There are two extra elements which are role and view when subjects access operations. The policy is described as follows:

After identity authentication, a subject is assigned to several roles, among which there are constraints called constraints1. Objects organized by views are used to call System resources, which benefits to fine-grained security modeling and management. There are constraints called constraints2 between operations of objects. The content of policy item is a mapping from roles to views. The mapping is realized by constraints called constraints3, the relationship between the elements is shown in the following Fig.2.

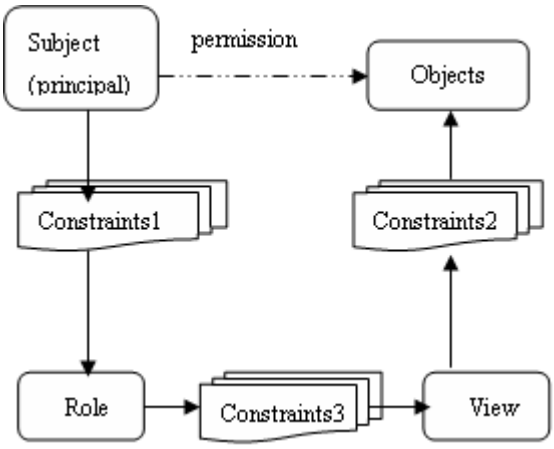

Fig. 2. Policy rules

There are three kinds of Constraints:

constraints1: There are three kinds of constraints when a role is defined: base number constraints; leading role constraints; mutual exclusion constraints.

constraints2: Leading operation constraints.

constraints3: View content constraints: including views that a role possesses when the role is initialized and roles that a view is assigned to when the view is defined.

- Authorization changes

The above authorized rules except role mutual exclusion constraints are deployed when the system is initialized. System also needs to dynamically assign or remove specific views according to security context in running time. We declare the definition of dynamic rules as follows:

dynamic submi1 **observes** Document

{ Submission
removes
view2 on this **from** Author }

When the operation Submission of Document is called, view2 would be removed from the current Subject who has the role of Author. The assignments and removals of views specified by a dynamic rules definition are carried out by a dedicated interceptor which observes successful returns from operations, and then dynamic authorization is carried out. To do so, the interceptor needs to know about all of the dynamic rules definitions that are applied to the object from which processing returns. This information can be retrieved during the initial session setup and stored in the session context. The process of dynamic authorization is shown in Fig.3: when principle requests for a specific operation, the interceptor firstly detects security context, if the operation is permitted to call and executed successfully, the module of interceptor will change security session context by dynamic rules.

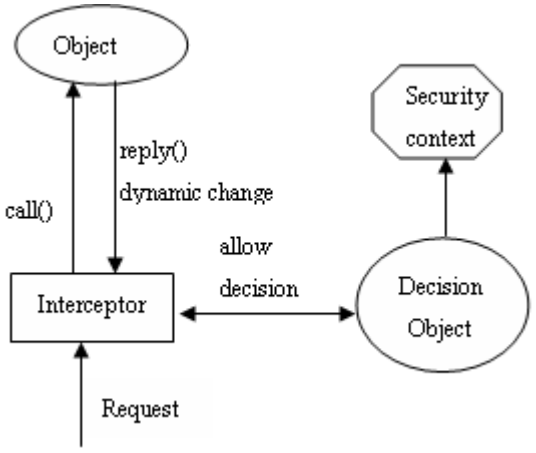

Fig. 3. Dynamic change authentication

3.2. Object management

System resources are encapsulated by objects and the visits to them are accomplished by calling operations. View, the subset of object operations, is used to realize fine-grained resource management. In the process of security modeling, views can be modeled by using the objects from system modeling. Next, we illustrate the modeling process. Assume that there are two types of resource: Folder and Document, and they respectively have the following operations:

```
interface Folder
{Lookup();
 Appending();
 listing();
 Removing(); }

interface Document
{Reading();
 Updating();
 Submission(); }
```

According to system analysis, a role called author who might use Lookup() and Append() operations of Folder as well as Reading(), Updating() and Submission operations of Document, therefore four of views can be established as follows.

View view1 control Folder restricted to author
{ Allow Lookup();

```
    Append(); }

View view2 control Document
{ Allow Reading();}

View view3:view2  restricted to author
{ Allow
      Updating();}
View view4 control Document restricted to author
{ Allow Submission();}
```

View1, view4 and view3 inherited from view2 are all restricted to author, therefore, a permission item is defined when a view with role constraints is defined. In the above definition, an operation is the smallest resource grain. The fine-grained authorization avoids the shortage of traditional access control in which there are only three coarse-grained operations to resources: reading, writing, execution, therefore it is easy to combine security modeling into system modeling.

3.3. Principal management

Technically, requests in a distributed system are always initiated by processes and arrive at a target over potentially insecure channels. Processes are commonly called subjects in the security literature, from the perspective of an access control mechanism at the target object side, the only immediately identifiable entity is the communication endpoint and thus the channel over which a request arrived. However, channels are a low–level, technical concept that is not suitable as a basis for access decisions. Consequently, other access control information needs to be obtained and authenticated, which then permits to deal with a more abstract notion of principals. A channel must present credentials, which are a proof that the channel speaks for a principal at this time.

The abstraction of principals is realized by supplying and verifying credentials and these credentials must be created and managed. Typically, credentials in distributed systems are public key certificates and rely on public key infrastructure. A typical way of assigning credentials in current distributed systems is to store them in repositories such as provided by the X.500 or LDAP directories. These support hierarchical modelling of organizations and thus allow managers to navigate the user population using structures with which they are familiar, i.e., organizational structures like branches, departments, and groups.

3.4. Matrix of security context

The access control policy ultimately consists of an access control matrix and dynamic authorization rules [3].Subjects and roles have separate rows in the matrix because assignment need to refer to individual subjects , not just to roles . Types have columns in the matrix because a resource is assigned to a principal on all objects of a given type. We illustrate as follows:

There are two type resources which are Folder and Document and three roles: Secretary, Author and Editor, all the roles are assigned operations respectively by views. When the subject---Zhang who is assigned roles of Secretary and Editor, login, there will be a new item to Zhang in the matrix. Because secretary role has the operations: Lookup() in Folder , Reading() in Document , and Editor role has the operations: Listing (),Append(), Removing() in Folder and Reading() in Document , the subject will have the following operations: Lookup(), Listing(), Append(), Removing() in Folder and Reading() in Document , as shown in Table 1.

Table 1. Matrix of security context

Type Principal	Folder	Document
Secretary	Lookup	Reading
Author	Lookup Appending	Reading Updating Submission
Editor	Listing Append Removing	Reading

zhang	Lookup Listing Appending Removing	Reading

The matrix supports a structure in context cache to realize fine-grained resources management and provides a reference for decision-making in security context.

The above model prevents active packets from calling unsafe operations by fine-grained object management and realizes flexible access control by the combination of dynamic and static authorization.

4. Policy deployment

The required access language must allow developers to provide a static policy specification that is independent of the target environment and is delivered with the application. This specification may be refined during deployment in the target environment and managed during the lifetime of the application.

Before deployed, an access policy needs to check type and verify language constraints. The final product of the policy development stage is a policy design document that is delivered as a descriptor file with the application. The XML is an established standard format for descriptors. The advantage of XML is that XML is an open standard for which a variety of tools is publicly available. Therefore, the compiler is designed as a combination of a pre-compiler and a backend verifier that accepts XML files as input, as shown in Fig.4.

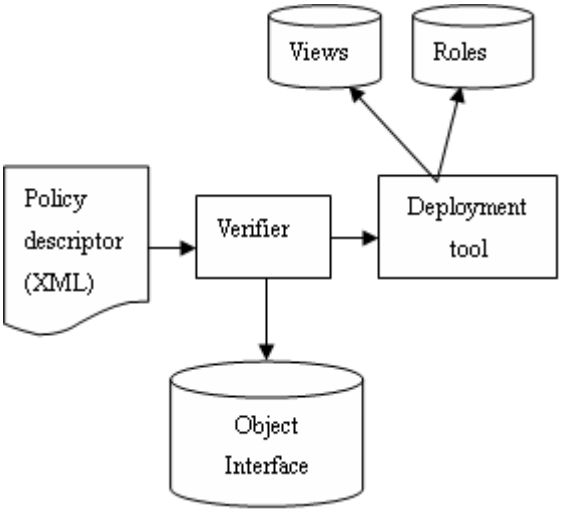

Fig. 4. Policy deployment

5. Conclusions

- The main question addressed in this paper is how the deployment and management of application–oriented access control policies in distributed object systems can be supported.
- The security management of this access control mechanism includes: policy management, objects management, principle management.
- The mechanism realize dynamic authentication according to security context.

Acknowledgements

The Project supported by China-Australia Special Fund for Scientific and Technological Cooperation (50810076) and also supported by SRF for ROCS, SEM. Thanks for their support.

References

[1] W.C. Richard, Role Activation Management in Role Based Access Control. ACISP, 2008.

[2] T. Ryutov, Access Control Framework for Distributed Applications. http://gost.isi.edu/info/gaaapi/doc/drafts/frmw_draft5.txt, November, 2005.

[3] D.F. Ferraiolo, Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security, 2004.

[4] K. L. Calvert, S. Bhattacharjee, E. W. Zegura, Directions in active networks. IEEE Communication Magazine, 1998.

[5] D. S. Alexnder, The Active Network Encapsulation Protocol (ANEP). http://www.cs.upenn.edu/~switcware/ANEP/docs/ANEP.txt, 2000.